



GETTING STARTED WITH THE LATEST IN PROGRAMMING MICROCONTROLLERS

■ BY CHUCK HELLEBUYCK

GETTING STARTED WITH THE PICBASIC™ PRO COMPILER AND MPLAB® IDE

EMAIL FEEDBACK HAS MADE IT CLEAR THAT there are many who would like me to go back and cover how to get started, as the title of the column suggests. Though I've covered this in many previous columns, much has changed, and there are still some beginner topics I have not covered.

One area missed is how to set up and use Microchip's MPLAB IDE for developing your programs. If you write in assembly, C, or even microEngineering Labs' PICBASIC or PICBASIC PRO compilers, you can use the same MPLAB IDE. This is important to learn, especially for those who may graduate from a hobbyist programmer to a professional programmer.

All Microchip tools work directly with the MPLAB IDE. This includes the PICkit™ 2 Starter Kit (Part #DV164120), which I've covered in many articles before and I plan to use again as the hardware. For the software, I decided to keep it simple and inexpensive by using the sample version of the PICBASIC PRO compiler. This is the Basic language compiler I've written about many times before. After all, the BASIC acronym stands for "Beginner's All-purpose Symbolic Instruction Code." The keyword, here, is beginner.

The sample version of the PICBASIC PRO compiler is available as a free download from <http://melabs.com/pbpdemo.htm>, and the MPLAB IDE is available as a free download from www.microchip.com/mplab. The PICkit 2 Starter Package will cost you \$49.99 plus shipping if you purchase it from www.microchipdirect.com, but you can get it for the same price from Mouser or any other catalog source. The starter package includes a PIC16F690 microcontroller (MCU), so the free compiler and IDE downloads (plus the starter package) will give you everything you need to start programming. If you want to follow along with this article, you'll need these items.

INSTALLATION

The MPLAB IDE is a very powerful tool, with a lot of features. I will go over the essentials you'll need to know in order to get started quickly.

First, download the latest release of the MPLAB IDE.

As I write this article, version 8.02 is the latest. The MPLAB IDE download is a zip file that needs to be extracted into the directory of your choice. Once you have un-zipped the files, you will see a file named "Install_MPLAB_v8.02.exe." This is the installation setup file that you need to run. Follow the installation procedure and let it install at the default directory. This will put the MPLAB IDE and all of its components in the C:\Program Files\Microchip directory of your hard drive.

Next, download the PICBASIC PRO compiler demo version into a directory on your hard drive. This includes an installation file called "PBPDEMO3.EXE." Run this file and let it install at the default directory, which will be located at C:\PBPDEMO.

Next, download the file called "PBPlugins.bat." This file is strictly for the MPLAB-PBPRO connection. You can get this program from the <http://melabs.com/support/mplab.htm> page, along with details on how to use the PICBASIC PRO compiler with the MPLAB IDE. I'm telling you my way of doing this because I found one little glitch that I could not get around, and I suspect you'll run into the same thing. Run the PBPlugins.bat program. The instructions at microEngineering Labs' [mplab.htm](http://melabs.com/mplab.htm) Web page explain how to update the search path for the version of Microsoft Windows® you are running. Follow those instructions and update the search paths.

Next is the trick I recommend for getting around the glitch I ran into. Go to the C:\PBPDEMO directory and you will see three folders — INC, MCS, and SAMPLES. Add

NOTE: The Microchip name and logo, MPLAB, and PIC are registered trademarks of Microchip Technology, Inc., in the USA and other countries. PICkit is a trademark of Microchip Technology, Inc., in the USA and other countries. All other trademarks mentioned herein are property of their respective companies.

a new directory called PFILES. Now, go to the directory where the MPLAB IDE is stored (C:\Program Files\Microchip\MPASM Suite) and you will see a bunch of files that start with a "P" and end with an ".inc." These include the files the compiler will look for and, for some reason, I could not get the MPLAB/PICBASIC PRO combination to find them. I even changed all the recommended path statements shown on microEngineering Labs' MPLAB instruction page and it still could not find them. You can copy all those Pxxx.inc files into the PFILES directory that you made, but I suggest you just copy the list below:

P12F683.INC	P16F871.INC
P16F84.INC	P16F872.INC
P16F84A.INC	P16F873.INC
P16F627.INC	P16F873A.INC
P16F627A.INC	P16F874.INC
P16F628.INC	P16F874A.INC
P16F628A.INC	P16F876.INC
P16F688.INC	P16F876A.INC
P16F690.INC	P16F877.INC
P16F870.INC	P16F877A.INC

This list includes all the MCUs supported by the PICBASIC PRO sample version, which will make it easier to find these files later when we create a project in the MPLAB IDE.

SETUP

Now that everything is installed, we need to set up the MPLAB IDE to recognize the PICBASIC PRO compiler.

- 1) Start MPLAB and select "Set Language Tool Locations" under the Project menu.
- 2) Select the "PICBASIC PRO Toolsuite" name.
- 3) Use the browse button to select PBPDEMOW.EXE in the PBPDEMO directory where the PICBASIC PRO Demo version was installed.

Figure 1 shows the window you should see when you complete these steps. By choosing this path, you are indicating to the MPLAB IDE where the PICBASIC PRO compiler is located on your hard drive. Click on the OK button to accept this, and you are now ready to use the MPLAB IDE with the PICBASIC PRO demo version. If you have the full version of PICBASIC PRO, all the steps are the same except the PICBASIC PRO will be installed in the C:\PBP directory, and the file you select in the browse window is PBPW.EXE.

FIRST PROJECT

Now, you are ready to build your first project. In the MPLAB IDE, all the software files you create will be connected by a project structure. The easiest way to do this is to use the Project Wizard utility

in the MPLAB IDE. Click on the MPLAB Project menu and select the Project Wizard. You will be walked through several windows, which take you through the following steps:

- 1) *Select a device.* Choose the PIC16F690, as that is the part included in the PICKit 2 Starter Kit.
- 2) *Select a language tool suite.* Choose the PICBASIC PRO tool suite.
- 3) *Create a new project.* Use the browse button, and select the directory where you want to store the project and all of the files. I suggest you create it as close to C: as possible, to keep the path name short. Figure 2 shows my project, entitled "16F690_Blink" in the PBPCode directory that I created at the root.
- 4) *Add existing files to your project.* This is where you may select one of the PICBASIC PRO sample files that you wish to use or modify, or maybe an older file that you wrote. For this example, select the BLINK.BAS file in the PBPDEMO/SAMPLES directory. Highlight it, and then click on the Add>> button.

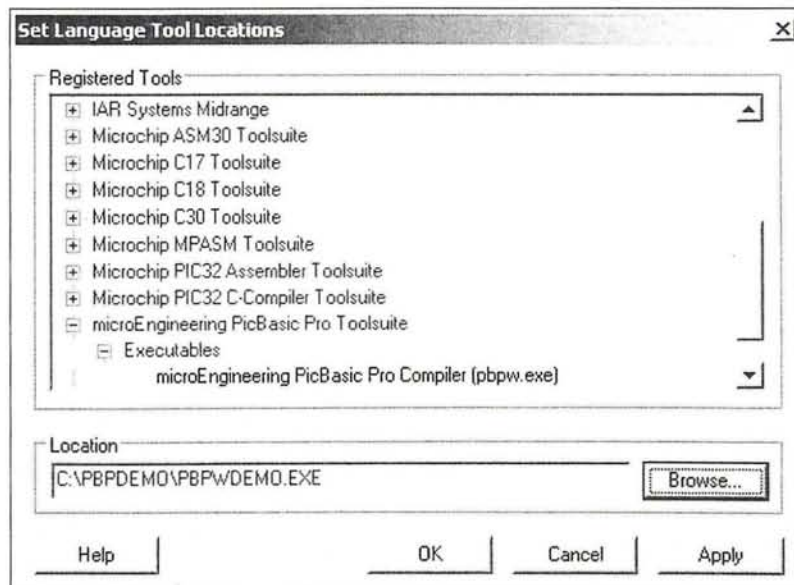
WAIT!!!! Don't press the "Next" button, yet. Instead, use a trick that I found — change to the PFILES directory you created and select the P16F690.INC file. Then, click the Add>> button to add it to the project, as well. This will save you an error later. Finally, next to each file, you will see a big "A." Click on that A until it changes to a C. This will automatically copy these files to your project directory. Figure 3 shows what your screen should look like.

- 5) At this point, you're done so click the FINISH button.

FIRST PROGRAM

Odds are that your MPLAB screen is now blank. If so,

■ FIGURE 1. PICBASIC PRO Language Selection.



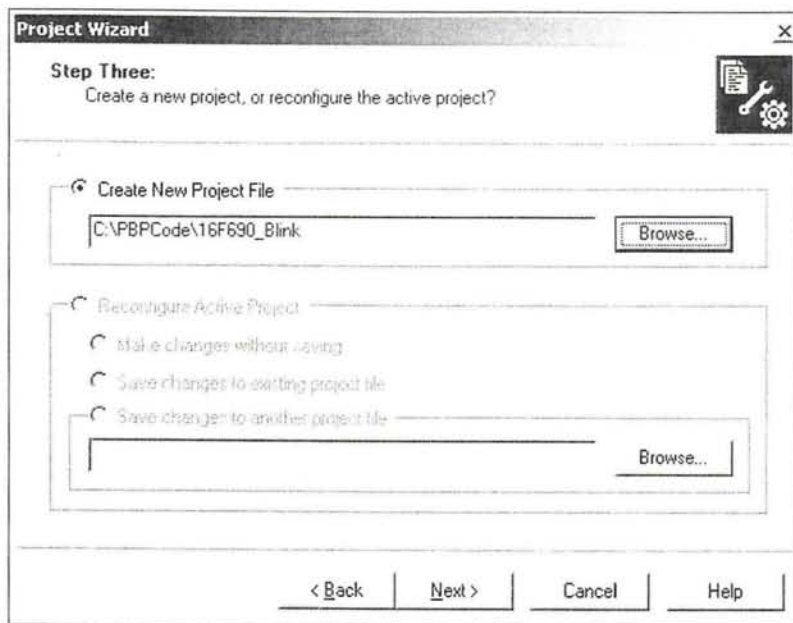
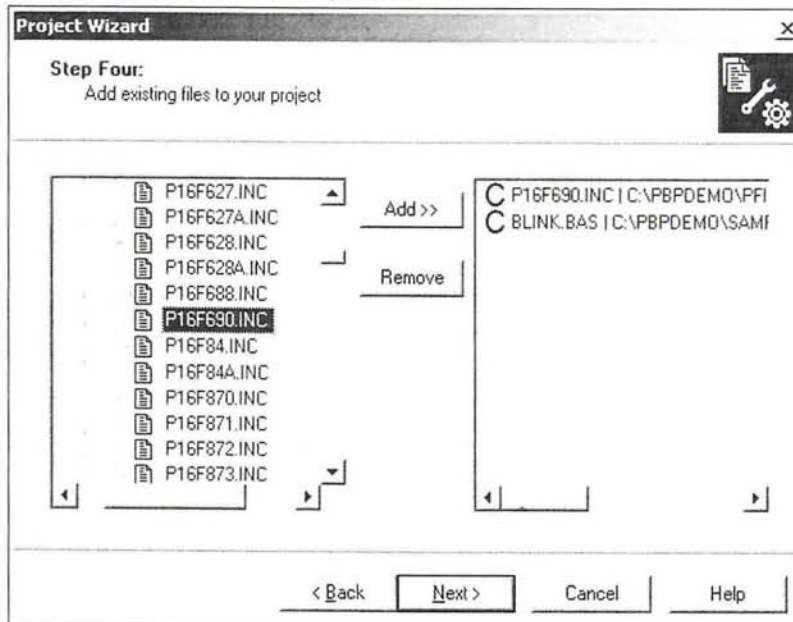


FIGURE 2. Step Three of Project Creation.

you need to click on the VIEW menu and select the PROJECT and OUTPUT windows. By clicking on these, a check mark will show up next to the menu selection, and the windows will appear in the MPLAB IDE with the project files shown in the Project window. The Output window will be blank. Figure 4 shows the final view. The BLINK.BAS and P16F690.INC files are listed and can be opened by double clicking on them. To actually run the BLINK.BAS file on the PIC16F690 MCU, we need to modify the basic file slightly. The sample file is written to work on the PORTB register, and we need to drive the PORTC pins. I modified the sample program to look like Listing 1. This is really a simple program, which makes it easier to prove out all the steps to get your first program

FIGURE 3. Step Four of Project Creation.



working. At the top of the program, though, are a few statements that might confuse the beginner. These are shown below:

```
ANSEL = 0      ' Initialize A/D ports off
CM1CON0 = 0    ' Initialize Comparator 1 off
CM2CON0 = 0    ' Initialize Coparator 2 off
```

These statements are required for using the PIC16F690 MCU's I/O as digital pins. The PIC16F690, like many other PIC MCUs, multiplexes the pin connections with other features. The PIC16F690 MCU has both Analog-to-Digital Converter (ADC) ports and comparators that share the actual pin connections with the digital I/O circuitry. To use the digital I/O pins, you must make sure the ADC and comparators are disconnected. For this project, we'll disconnect them by clearing the bits in these registers — Analog Select Register (ANSEL), Comparator 1 Control Register (CM1CON0), and Comparator 2 Control Register (CM2CON0).

The rest of the program is just a High, Low, and Pause loop that acts on the PORTC pin, RC0. This pin is connected to the DS1 LED on the PICKIT 2 starter board. By flipping the level on the RC0 pin from high to low with a pause in between and looping through that sequence multiple times, we make the LED flash.

Once the program is written, simply press the F10 button to compile the PICBASIC PRO file into a binary .hex file. If everything compiles without errors, you will see a "Build Succeeded" message in the output window. If you receive an error message, it will tell you in which line of code it is so that you can see what typo you may have accidentally made.

PROGRAMMING THE PIC16F690 MCU

Now that we have a binary .hex file, we need to load it into the PIC16F690 MCU so that it can run. For this, I will use the PICKIT 2 Starter Kit (see Figure 5), with the PICKIT 2 Programmer connected to the PC's USB port. The PICKIT 2 Programmer will then power the development board from the USB port and program the PIC16F690 MCU when inserted into its socket through the board's programming connector. Because the MCU is connected to the LEDs, switch, and potentiometer on the development board, we will use the In-Circuit Serial Programming™ feature to download the .hex file into the PIC16F690 MCU. This just means that you don't have to remove the MCU in order to program it.

To move forward from this point, we need to connect the PICKIT 2 Programmer to the USB port and then enable the programmer in the MPLAB IDE. We do so by selecting the PICKIT 2 from the "Programmer" menu at the top of the MPLAB screen (see Figure 6).

FIGURE 4. The MPLAB IDE Window.

The output window will gain a PICKit 2 tab and show the status of the PICKit 2 Programmer. You should see the following text displayed in the output window:

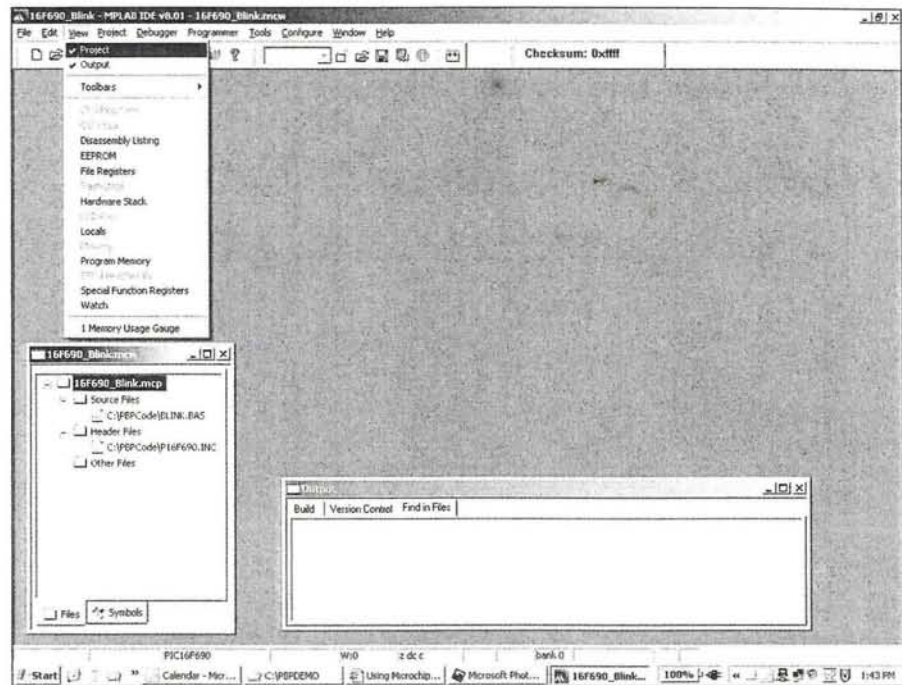
```
Found PICKit 2 - Operating System
Version 2.20.0
Target power not detected - Powering
from PICKit 2
PIC16F690 found (Rev 0x4)
PICKit 2 Ready
```

The message will indicate the operating system in the PICKit 2 Programmer, where the development board is receiving power from, and which MCU it detected. After this, a message stating the PICKit 2 is ready to program will be displayed. At the top right of the MPLAB window, the control buttons for the PICKit 2 Programmer should also appear, as shown in Figure 7. These buttons allow you to:

- 1) Program the complete part.
- 2) Read program memory.
- 3) Read EEPROM.
- 4) Verify that the program inside matches what you programmed.
- 5) Erase the complete device.
- 6) Verify that the complete device is erased.

You also have control over the MCLR reset line on the MCU, which are the rising- and falling-edge icons. By clicking on the rising-edge icon, you allow the PIC16F690 MCU to run the program. The last icon is a miniature PICKit 2, which just allows you to re-check the status of the programmer.

To load the BLINK.hex file into the PIC16F690 MCU, simply click on the first icon button and the PICKit 2 Programmer will handle the rest. You will see the status in the PICKit 2 output window. It will first erase the



MCU, and then program and verify it. After this, the programmer is ready to run. Click on the rising-edge button to bring MCLR to VDD. The LED should start to flash.

ERRORS

To get through all of this without errors is a very good start. However, chances are you might see a few errors, or get all the way to the end and find that the LED does not flash. I'll try to cover a few of the more common errors that the beginner might run into. The first involves the PICBASIC PRO compiler and the MPLAB IDE/Windows structure. For some reason, no matter how I changed the path structure in Windows or reset things in the MPLAB setup screens, I would encounter the error shown in Figure 8 when I first tried to run the PICBASIC PRO compiler in the MPLAB IDE.

In fact, I received a whole list of errors that started with the line "Cannot open file ... P16F690.INC." This is

LISTING 1: BLINK.BAS Sample Program

* Example program from manual to blink and LED connected
* to PORTC.0 about once a second.

```
ANSEL = 0          * Initialize A/D ports off
CM1CON0 = 0        * Initialize Comparator1 off
CM2CON0 = 0        * Initialize Comparator2 off

loop: High PORTC.0  * Turn on LED connected to PORTC.0
    Pause 500       * Delay for 5 seconds

    Low PORTC.0     * Turn off LED connected to PORTC.0
    Pause 500       * Delay for 5 seconds

    Goto loop       * Loop back and blink LED forever

End
```



FIGURE 5. PICKit 2 Starter Kit.

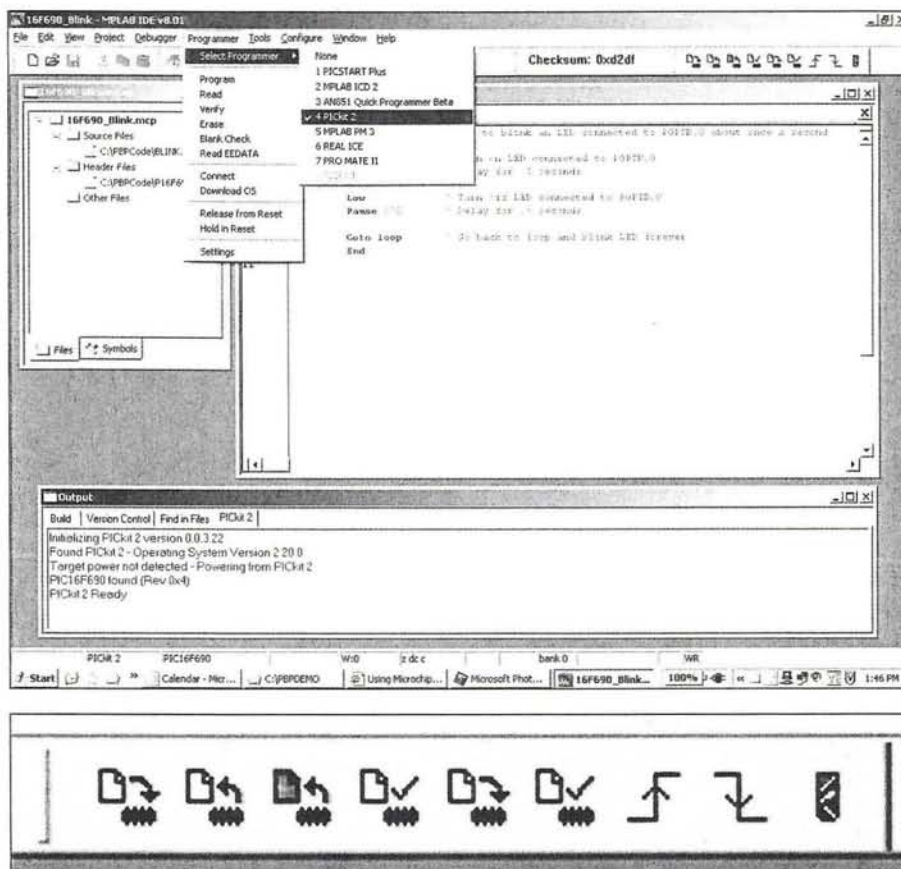


FIGURE 6. PICkit 2 Programmer Selection.

MCU you are using. In this case, the file is named 16F690.inc. You will find it in the PBP or PBPDEMO directory, where you installed the PICBASIC PRO compiler. The file will contain a `_config` line, like the example below:

```
_config _INTRC_OSC_NOCLKOUT & _WDT_ON
& _MCLR_ON & _CP_OFF
```

This line in the 16F690.inc file is where the PICBASIC PRO compiler gets the information on how to set the configuration bits inside the .hex file. In this example, the internal RC oscillator is used as the system clock. This is what I recommend for the PIC16F690 MCU, but you can change it to an external oscillator if you need more accuracy. For the beginner, I would not worry about all of this — just know that it exists. However, if you find that your LED does not flash, then you might want to make sure the settings are adequate for what you need. For example, if you are developing on a

board that has an external 20 MHz crystal and you keep finding that the program is running slow, you might have the internal oscillator set up in the configuration.

One of the biggest errors I've seen with beginners is the exact opposite—they think they are using the internal oscillator, but the configuration is set to run from an external oscillator (i.e., `_XT_OSC`). The MCU won't run without a clock.

CONCLUSION

I covered a lot of ground in this article. However, if you use this setup and get that first LED to blink, you are ready to start creating more software programs without having to worry about all of the hardware setup connections. What I suggest you do is modify the PAUSE command value to get the LED to flash faster or slower. Then, perhaps try to duplicate the code and get it to flash a second LED. If you alternate the high and low commands, you can make the LEDs flash back and forth, like lights at a railroad crossing.

Some readers have shared that they think my column is good, but they sometimes get lost trying to follow along.

They feel I'm writing to a bunch of engineers, rather than to hobbyists. I understand this completely, however this is a normal reaction as my subject matter can seem complicated to

why I suggest you use the project wizard the first time you create a project, and include the P file for the part you are using to prevent this error from occurring. If you forget, you can add a copy of the P file later, but it has to be put into the same directory as the .bas file you created.

Another beginner error that crops up often involves the configuration settings. Outside the structure of your program, the PIC16F690 MCU has certain bits that are set at program time to control the watchdog timer, the power-up timer, the oscillator selection, and more. All of the options for the part can be seen by clicking on the `Configure>Configuration bits` menu selection in the MPLAB IDE (see Figure 9). You can manually select the options, or click on the little box in the upper-left corner to allow the compiler to set these in code.

I recommend that you set the options in code, because they will then be embedded in the .hex file used to program the MCU. The PICBASIC PRO compiler puts that configuration setup in a separate file that it calls at compile time. The setup will be in an .inc file that has the name of the

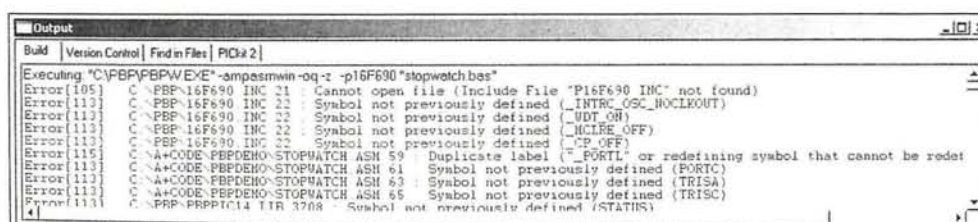


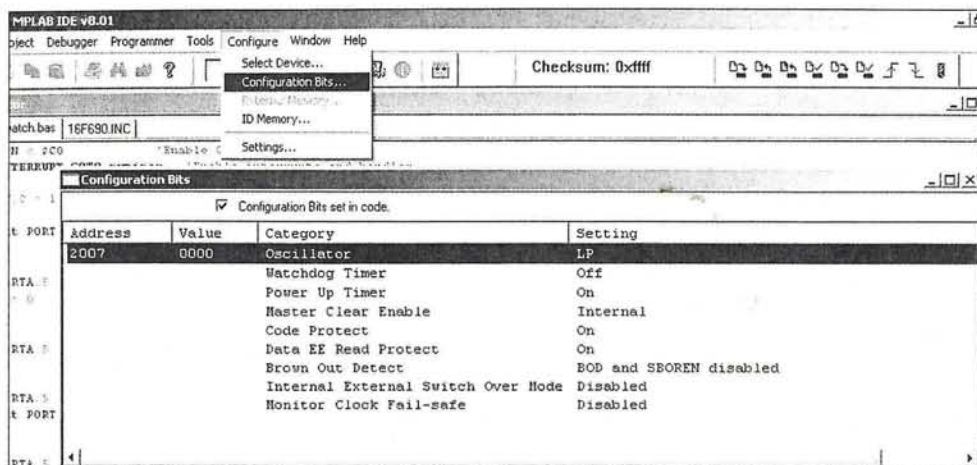
FIGURE 8. Compiler Error.

E 9. Configuration Menu.

people who are just getting started with MCUs. Once you get an LED to flash, the fear subsides and you begin to understand things much easier. You can go back and re-read my previous columns, and suddenly all of the complicated stuff becomes easier.

Another problem is that some readers are familiar with using little Basic modules that have all the inner details hidden from the user, in order to make things easier. You pay for that, though, by not having the ability to use all the features an MCU can offer. Many times, you sacrifice speed and memory space for simplicity, not to mention paying a lot more for the MCU. This will seem like a leap at first but, trust me, it's not that tough.

On the other side of the fence, I receive other comments stating that I often use too simple of an example — such as flashing an LED — to show how to get started (as I've done in this column). I will cover more complicated projects using this same MPLAB IDE, PICBASIC PRO compiler (sample version), and PICkit 2 Starter Kit setup in future columns. The idea I have with this new approach to the beginner path is to use a common, but powerful and professional set of development tools and software to create a step-by-step guide to getting started in programming. I plan to use this same setup in many future columns to remain consistent. I hope you'll continue to tune in. After readers get more comfortable with the PICBASIC PRO compiler, they can then advance to the full version or possibly convert to the C language. By then, my *Beginner's Guide to Embedded C Programming*



book should be in print, and I can help you down that path.

Please send your feedback on this particular article to me, so that I can determine how successful you were in getting all of this to work. My email address is chuck@elproducts.com. I try to answer all emails, but I sometimes find messages from readers caught in my spam filter. Please write "Nuts & Volts" or "N&V" in the subject line to help me find your email. Your feedback will enable me to explain the subject matter of my columns in more complete ways (there is only so much I can fit into a few *Nuts & Volts* pages).

Additionally, if you get a chance, check out my new web-site dedicated to my **books-www.elproducts.com**. If you are a fan of my modules and other hardware, you can now buy them from my friends at **www.beginnerelectronics.com**.

I hope you tune in to my next column (July '08 issue), where I'll show you how to use an ADC to read the potentiometer on the PICkit 2 development board. *NW*

CONTACT THE AUTHOR

■ Chuck Hellebuyck can be reached at chuck@elproducts.com.

Order online at:
www.melabs.com

Development Tools for PIC® MCUs
microEngineering Labs, Inc.

Phone: (719) 520-5323
Fax: (719) 520-1867
Box 60039
Colorado Springs, CO 80960

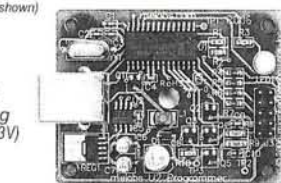
USB Programmer for PIC® MCUs

\$89.95 (as shown)

RoHS
Compliant

Programs PIC
MCUs including
low-voltage (3.3V)
devices

Includes
Software for
Windows
98, Me, NT,
XP, and Vista.



With Accessories for \$119.95:
Includes Programmer, Software, USB Cable,
and Programming Adapter for 8 to 40-pin DIP.

LAB-X Experimenter Boards



Pre-Assembled Board
Available for 8, 14, 18, 28,
and 40-pin PIC® MCUs
2-line, 20-char LCD Module
9-pin Serial Port
Sample Programs
Full Schematic Diagram

Pricing from \$79.95 to \$349.95

PICPROTO™ Prototyping Boards



Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for Most PIC® MCUs
Documentation and Schematic

Pricing from \$8.95 to \$19.95

BASIC Compilers for PICmicro®

Easy-To-Use BASIC Commands
Windows 98/Me/2K/XP/Vista

PICBASIC™ Compiler \$99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PICBASIC PRO™ Compiler \$249.95
Supports most PICmicro® MCU Families
Direct Access to Internal Registers
Supports In-Line Assembly Language
Interrupts in PICBASIC and Assembly
Built-In USB, I2C, RS-232 and More
Source Level Debugging

See our full range of products, including
books, accessories, and components at:
www.melabs.com



**EPIC™ Parallel
Port Programmer**
starting at \$59.95